IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

First Named Inventor :	khosro Khakzadi	Appeal No. ---
Appln. No. :	10/719,673	
Filed :	November 21, 2003	Group Art Unit: 2179
For :	CHIP DESIGN COMMAND PROCESSOR	Examiner: WIENER, Eric. A.
Docket No.:	03-1862/L13.12-0251	

BRIEF FOR APPELLANT

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

FILED ELECTRONICALLY
March 11, 2009

Sir:

This is an appeal from a Final Office Action dated July 9, 2008 and from a Pre-Appeal Panel Decision dated February 11, 2009 in which claims 1, 5-16, and 19-28 were finally rejected.

REAL PARTY IN INTEREST

LSI CORPORATION, a corporation organized under the laws of the state of California, and having offices at 1621 BARBER LANE, MILPITAS, CALIFORNIA 95035, has acquired the entire right, title and interest in and to the invention, the application, and any and all patents to be obtained therefor, as set forth in the Assignment filed with the patent application and recorded on Reel 020548, frame 0977.

RELATED APPEALS AND INTERFERENCES

There are no known related appeals or interferences which will directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

STATUS OF THE CLAIMS

I.	Total number of claims in the application.		
		Claims in the application are:	23
II.	Status of all the claims.		
	A.	Claims cancelled:	2-4, 17 and 18
	B.	Claims withdrawn but not cancelled:	none
	C.	Claims pending:	1, 5-16, and 19-28
	D.	Claims allowed:	none
	E.	Claims rejected:	1, 5-16, and 19-28
	F.	Claims Objected to:	none
III.	Claims on appeal		
		The claims on appeal are:	1, 5-16, and 19-28.

STATUS OF AMENDMENTS

An Amendment after Final was filed on August 12, 2008. The Amendment after Final was entered by the Examiner, as indicated by the Advisory Action mailed on August 27, 2008. However, the Examiner asserted that the Amendment did not place the application in condition for allowance.

SUMMARY OF CLAIMED SUBJECT MATTER

All citations to the specification refer to the page and paragraph numbers of the published application no. 2005/0114818, which was published on May 26, 2005.

A. Independent Claim 1 (Command Processor)

Claim 1 recites a command processor stored on a computer readable memory for use with a computer system. ("command processor 220" in FIG. 2; "command processor 410" in FIG. 4, p. 1, paragraph [0012], and p. 2, paragraphs [0027] to [0028]). The command processor includes a graphical user interface program for providing a graphical interface to the computer system and includes a command interpreter. ("command processor", "GUI", and "command interpreter" in FIGS. 2 and 4). The command interpreter loads one or more configuration commands into the command processor from at least one of a user specified command configuration script comprising the one or more configuration commands or from a command line interface in which the one or more configuration commands are entered by a user, interprets the configuration commands and modifies the graphical user interface at run time of the graphical user interface according to the interpreted configuration commands. (p. 2, paragraph [0028] to p. 3, paragraph [0033], p. 4, paragraphs [0045] to [0047], and FIGS. 10A and 10B). The command interpreter interprets and modifies by building graphical objects within the graphical interface according to the interpreted configuration commands, assigning functionality to the built graphical objects according to the interpreted configuration commands, and displaying the graphical objects within the graphical interface according to the interpreted configuration commands. (p. 4, paragraphs [0046] to [0047], p. 6, paragraphs [0067] to [0068]). The graphical objects are selectable by a user to access the assigned functionality to produce an integrated circuit design. (p. 6, paragraphs [0067] to [0068]).

The subject matter of claim 1 is fully supported by the application as filed, including at least FIGS. 2, 4, 5, 7, 9, 10A, and 10B, and including the specification at least at page 2, paragraph [0027] to page 3, paragraph [0033], p. 4, paragraphs [0045] to [0047], and p. 6, paragraphs [0067] to [0068].

B. Independent Claim 8 (Method of providing a fully customizable GUI)

Claim 8 recites a method of providing a fully customizable graphical user interface. (p. 1, paragraph [0013] to p. 2, paragraph [0014], p. 2, paragraphs [0027] to p. 3, paragraph [0033], p. 4, paragraphs [0046] to [0047] and p. 6, paragraphs [0067] to [0068]). Upon execution of a command processor by a user, the method includes loading a top level Tool Command Language (TCL) command into a namespace (p. 6, paragraph [0067]), where the command processor includes a graphical user interface (GUI) without graphical objects. (FIGS. 2 and 4, p. 4, paragraph [0047], and original claim 19). The method further includes loading one or more TCL commands into the command processor from a command line in which the one or more TCL configuration commands are entered by the user (p. 1, paragraph [0013] to p. 2, paragraph [0015] and p. 2, paragraph [0030]), building graphical objects according to the TCL configuration commands (p. 2, paragraphs [0014] to [0015], p. 4, paragraphs [0046] to [0047], and p. 6, paragraphs [0067] to [0068]), and assigning functionality to the built graphical objects according to the TCL configuration commands (p. 1, paragraph [0013], p. 2, paragraphs [0028] to [0031], p. 3, paragraph [0033], p. 4, paragraphs [0046] to [0047], and p. 6, paragraphs [0067] to [0068]). The method also includes displaying the graphical objects within the GUI according to the TCL configuration commands, where the graphical objects selectable by the user to produce an integrated circuit design. (p. 3, paragraph [0033], and p. 4, paragraph [0047]).

The subject matter of claim 8 is fully supported by the application as filed, including at least FIGS. 2, 4, 5, 7, 9, 10A, and 10B, and including the specification at least at page 1, paragraph [0013] to page 2, paragraph [0015], page 2, paragraph [0027] to page 3, paragraph [0033], p. 4, paragraphs [0045] to [0047], and p. 6, paragraphs [0067] to [0068].

C. Independent Claim 19 (Method of Providing a GUI)

Claim 19 recites a method of providing a graphical user interface includes loading a top level Tool Command Language (TCL) command into a namespace upon execution of a command processor (p. 1, paragraph [0013] to p. 2, paragraph [0014], p. 2, paragraphs [0027] to p. 3, paragraph [0033], p. 4, paragraphs [0046] to [0047] and p. 6, paragraphs [0067] to [0068]) and providing a command interpreter for interpreting one or more configuration commands from a user. (pp. 1-2, paragraph [0014], p. 2, paragraphs [0026] to [0029], p. 4, paragraphs [0046] to [0047], and p. 6, paragraphs [0067] to [0068]). The method further includes loading a configuration command of the one or more configuration commands into the command processor from at least one of a user specified command configuration script comprising the configuration command or from a command line in which the configuration command is entered by the user. (p. 1, paragraph [0013] to p. 2, paragraph [0015] and p. 2, paragraph [0030]). The method also includes assembling a graphical user interface having no hard coded objects (FIGS. 2 and 4, p. 4, paragraph [0047], and original claim 19) and including at least one graphical user interface (GUI) component based on interpreted configuration commands from the user, where the at least one graphical user interface (GUI) component is selectable by a user to access an associated function to generate an integrated circuit design, where all objects within the graphical user interface are user defined through the one or more configuration commands. (p. 1, paragraph [0013], p. 2, paragraphs [0028] to [0031], p. 3, paragraph [0033], p. 4, paragraphs [0046] to [0047], and p. 6, paragraphs [0067] to [0068]).

The subject matter of claim 19 is fully supported by the application as filed, including at least FIGS. 2, 4, 5, 7, 9, 10A, and 10B, and including the specification at least at page 1, paragraph [0013] to page 2, paragraph [0015], page 2, paragraph [0027] to page 3, paragraph [0033], p. 4, paragraphs [0045] to [0047], and p. 6, paragraphs [0067] to [0068].

D. Independent Claim 24 (Integrated Circuit Software Design Suite)

Claim 24 recites an integrated circuit software design suite stored on a computer readable memory (FIGS. 2 and 4, p. 2, paragraphs [0015] and [0026] and [0032], p. 3, paragraphs [0034], [0037] to [0044], p. 4, paragraph [0050] to [0051] and [0054], p. 5, paragraphs [0055] to [0056] and [0064] to [0065], and p. 6, paragraphs [0066] and [0069]). The integrated circuit software design suite includes a command processor having a graphical user interface and a command interpreter for interpreting user commands. ("command processor", "GUI", and "command interpreter" in FIGS. 2 and 4). The graphical user interface is specified entirely by a user via a user input including one or more configuration commands provided to the command processor at run time of the command processor and interpreted by the command interpreter, where the configuration commands build graphical objects within the graphical user interface and assign functionality to the built graphical objects. (p. 2, paragraph [0015], p. 4, paragraph [0047], and p. 6, paragraph [0068]). One or more design tools correspond to processes within an integrated circuit design process. (FIGS. 4-9, p. 2, paragraph [0015], p. 4, paragraph [0047], and p. 6, paragraph [0068]. The one or more design tools operate under control of the command processor and within the graphical user interface. (FIG. 4, p. 4, paragraphs [0046] to [0047], p. 6, paragraphs [0067] to [0068]).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 8-16 are allowable over the asserted combination of U.S. Patent No. 5,974,253 ("Nahaboo") and U.S. Patent No. 5,493,508 ("Dangelo"), at page 3, paragraph 5 of the Final Office Action;

B. Claims 24-28 are allowable over the asserted combination of Nahaboo and Dangelo;

C. Claims 1 and 5-7 are allowable over the asserted combination of Nahaboo and Dangelo; and

D. Claims 19-23 are allowable over the asserted combination of Nahaboo and Dangelo.

ARGUMENT

This communication is in response to the Final Office Action mailed on July 9, 2008 in which claims 1, 5-16, and 19-28 were pending. All of the pending claims are in condition for allowance. Reconsideration and notice to that effect is respectfully requested.

A. CLAIMS 8-16 ARE ALLOWABLE

Appellant respectfully traverses the rejection of claims 8-16 over the asserted combination of U.S. Patent No. 5,974,253 ("Nahaboo") and U.S. Patent No. 5,493,508 ("Dangelo"), at page 3, paragraph 5 of the Final Office Action. In particular, with respect to claim 8, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest "upon execution of a command processor by a user, loading a top level Tool Command Language (TCL) command into a namespace, the command processor including a graphical user interface (GUI) without graphical objects," "loading one or more TCL commands into the command processor from a command line in which the one or more TCL configuration commands are entered by the user," "building graphical objects according to the TCL configuration commands," "assigning functionality to the built graphical objects according to the TCL configuration commands", and "displaying the graphical objects within the GUI according to the TCL configuration commands, the graphical objects selectable by the user to produce an integrated circuit design", as recited in claim 8.

In general, the Office blurs a distinction between the interface development tool of Nahaboo and the associated application with the embedded WOOL interpreter program that actually implements the developed (designed) user interface file. In particular, Nahaboo discloses an interface design tool that can be used by a software developer to produce a user interface, which can be packaged with a software application (after it is compiled) so that the user interface is invoked and used in connection with a software application via the interpreter program. However, the developer of the user interface and the associated application may not be the end user. Accordingly, though the design tool of Nahaboo allows a developer to alter the user interface during design, there is no suggestion or teaching that the user interface can be modified during operation by an end user (who is an integrated circuit designer and <u>not</u> the software developer). By ignoring that distinction, the Office then makes the extension that the interface development tool could be incorporated into the circuit design tool of Dangelo.

Nahaboo discloses a user interface description tool that uses an interpreted language where both data and the program have similar representations and that uses libraries of command objects and graphical objects. *See Nahaboo*, Abstract. Nahaboo notes that poor interface designs can result in mitigated commercial success. *See Nahaboo*, col. 1, lines 25-26.

Nahaboo discloses that "the purpose of this invention is to define an <u>extremely flexible interface development tool</u> that can be used regardless of the application." *See Nahaboo*, col. 1, lines 29-34 (emphasis added). Nahaboo discloses that the interface development tool uses a list processing language (LISP) type of interpreted language that can be interpreted by an interpreter (WOOL) language program. *See Nahaboo*, col. 1, lines 35-39. Further, Nahaboo discloses that the interface development tool produces an interactive assembly geometry specification that is translated to a file that can be used by an application and stored on a disk, and that the file contains the user-interface format in the form of a WOOL language program. *See Nahaboo*, col. 2, lines 10-14.

Nahaboo states that the interactive interface tool allows the user to create and modify interfaces and windows that exhibit behaviors that can be easily duplicated. *See Nahaboo*, col. 4, lines 39-60. However, Nahaboo discloses an interface development tool to design and refine interfaces that can be used with applications, but does not suggest that the user of the applications allow for design and refinement of the interface "at run time of the graphical user interface," as recited in claim 1. Instead, Nahaboo discloses that the graphical user interface can be designed and edited and then stored for use with an application. At runtime of the application, the saved interface is loaded and executed using a WOOL program language interpreter that is embedded with the application. *See Nahaboo*, col. 1, lines 35-39. Thus, the WOOL interpreter program, **not the interface development tool,** is embedded with the application. While Nahaboo discloses that the interface description program can be expanded by adding other "widget" classes, Nahaboo discloses that "these classes can be added by describing the new attribute types of each class in the WOOL language so that the new "widgets" can be edited, for example, using the menu of the interface development tool. *See Nahaboo*, col. 2, lines 21-56. In Nahaboo, since the WOOL interpreter program loads and interprets the WOOL language program, such modifications or additions would take effect only upon restarting the application. Thus, Nahaboo discloses modification of the graphical user interface using an interface development tool, storage of the modified graphical user interface, and later execution using a WOOL interpreter program that is embedded with the application and that executes the stored graphical user interface file in conjunction with the application.

Appellant notes that Nahaboo discloses editing an interface using a separate application from that with which the interface is to be used. The edited interface is stored in a WOOL language program and the WOOL language interpreter is embedded with the application to interpret the interface file. However, Nahaboo discloses that the interface development tool allows for modification and design of interfaces, which are stored in a WOOL language program together with a WOOL interpreter program embedded within the application. *See Nahaboo*, col. 1, lines 35-39. The interface development tool of Nahaboo is not also used to produce an integrated circuit design. Rather, a separate application is executed that uses the GUI designed by the interface development tool of Nahaboo to display user selectable object. Nahaboo provides no suggestion or motivation for combining the interface development tool with the actual application that is executed by the user (and not the designer). Further, Nahaboo provides no suggest or teaching that the "loading one or more TCL commands into the command processor from a command line in which the one or more TCL configuration commands are entered by the user," "building graphical objects according to the TCL configuration commands," "assigning functionality to the built graphical objects according to the TCL configuration commands", and "displaying the graphical objects within the GUI according to the TCL configuration commands, the graphical objects selectable by the user to produce an integrated circuit design", as recited in claim 8. Thus, Nahaboo does not disclose all of the elements of claim 8.

Dangelo does not overcome the deficiencies of Nahaboo. Instead, Dangelo discloses a "methodology for generating structural descriptions of complex digital devices from high-level descriptions and specifications," which "uses as systematic technique to map and enforce consistency of the semantics..." *See Dangelo*, Abstract. Further, Dangelo discloses "a Graphical User Interface (Graphical UI) 806 facilitates user interaction with the CDE by: abstracting out those steps of the [integrated circuit] design flow that do not require the designer's intervention, assisting and guiding the designer through the various stages of the design process as outlined by the synthesis framework, and assisting the designer in the composition of the constraints file for optimization." *See Dangelo*, col. 16, lines 35-42. In Dangelo, the building blocks of the circuit, such as wire traces, mega-cells, mega-function blocks, and the like are chosen from pre-designed building block libraries. *See, e.g., Dangelo*, col. 11, lines 1-10. Dangelo fails to disclose or suggest "building graphical objects within the graphical interface according to the interpreted configuration commands," as recited in claim 8. Further, Dangelo does not disclose or suggest "upon execution of a command processor, loading a top level Tool Command Language (TCL) command into a namespace, the command processor including a graphical user interface (GUI) without graphical objects," as recited in claim 8. Instead, Dangelo discloses a graphical user interface with a graphical map. *See Dangelo*, Figure 8. Hence, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest all of the elements of claim 8, or of claims 14-16, at least by virtue of their dependency from claim 8.

In particular, Dangelo does not disclose a "command processor including a graphical user interface (GUI) without graphical objects," and does not disclose "loading one or more TCL commands into the command processor from a command line in which the one or more TCL configuration commands are entered by the user," "building graphical objects according to the TCL configuration commands," "assigning functionality to the built graphical objects according to the TCL configuration commands", and "displaying the graphical objects within the GUI according to the TCL configuration commands, the graphical objects selectable by the user to produce an integrated circuit design", as recited in claim 8.

Accordingly, the asserted combination of Nahaboo and Dangelo is missing at least one element of claim 8, and of claims 9-16, at least by virtue of their dependency from allowable claim 8.

Further, the Appellant submits that the asserted combination of Nahaboo and Dangelo constitutes an impermissible hindsight reconstruction using the present application as a template to piece together elements from Nahaboo and Dangelo. The fact finder must be aware of distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning. *See KSR Int'l Co. v. Teleflex Inc., citing Graham v. John Deere*, 383 U.S., at 36. **"Determination of obviousness cannot be based on the hindsight combination of components selectively culled from the prior art to fit the parameters of the patented invention**." ATD Corp. v. Lydall, Inc., 159 F.3d 534, 48 USPQ2d 1321 (Fed. Cir. 1998) (emphasis added); *see also KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. ___ (2007), *citing Monroe Auto Equipment Co. v. Heckethorn Mfg & Supply Co.*, 332 F.2d 406, 412 (CA6 1964)) (warning against a "temptation to read into the prior art the teachings of the invention in issue"). This impermissible hindsight reconstruction is present here, where the Office ignores the explicit teachings of Nahaboo relating to separate applications including 1) an interface development tool to construct user interfaces and 2) an application with an embedded WOOL interpreter program to utilize the created interfaces to produce an interface for use with the application. The design tool is to be used by a software designer, and not an end user. Further, the WOOL interpreter program is not indicated to allow a user to specify or control the interface, but rather Nahaboo discloses that the interface development tool is used to modify the interface. The interface development tool is indicated to be usable with a variety of applications, specifically because the WOOL interpreter program is embedded with each of the applications to load the interface file. Notwithstanding the suggestion of the Office that the design application of Dangelo could be combined with the interface development tool, the teachings of Nahaboo teach away from such a combination. The only motivation for modifying either Nahaboo or Dangelo to provide the elements disclosed in the claims is provided by the present application.

Further, the Office is reminded that rejections on obviousness cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *See KSR Int'l Co. v. Teleflex Inc., citing In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006). The Office suggests that because Nahaboo discloses "an extremely flexible development tool that can be used regardless of the application (column 1, lines 29-34)", then "Dangelo would thus look to Nahaboo regarding features of modifying a graphical user interface to produce a design." *See the Final Office Action*, p. 5, paragraph 5. Further, the Office asserts that Dangelo discloses that the interface of Dangelo may be specified for different users. *See the Final Office Action*, p. 5, paragraph 5. However, the Office misconstrues the teaching of Dangelo, which discloses a design compiler that enables the user to follow the process of the design from one environment to another via script shells and command line and which discloses a GUI interface that facilitates user interactions by abstracting out those steps in the design flow that down require the designer's intervention. *See Dangelo*, p.col. 16, lines 16-42. Dangelo does not disclose or suggest a user specified GUI. Further, while Nahaboo recognizes a problem with poor interface design, Nahaboo does not disclose or suggest the interface design tool be combined with the application. Rather, Nahaboo specifically discloses that the created interface is stored in a file with the application and that an interpreter is embedded with the application to read and implement the interface file. *See Nahaboo*, col. 1, lines 35-39, col. 2, lines 10-14 and 21-56. Nahaboo discloses that the design tool is used to modify the user interface, and Neither Nahaboo nor Dangelo provide any suggestion or motivation to incorporate an interface design tool with an application that generates integrated circuit designs.

Further, Nahaboo provide no suggestion or motivation for incorporating its interface design tool into the application, but rather teaches away from such incorporation by teaching embedding of a WOOL interpreter program instead. Accordingly, there exists no motivation for making the asserted combination. Therefore, the asserted combination of Nahaboo and Dangelo constitutes an impermissible hindsight reconstruction, and the rejection of the pending claims over the asserted combination of Nahaboo and Dangelo should be withdrawn.

B. CLAIMS 24-28 ARE ALLOWABLE

Appellant respectfully traverses the rejection of claims 24-28 over the asserted combination of Nahaboo and Dangelo. In particular, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest "the graphical user interface specified entirely by a user via a user input including one or more configuration commands provided to the command processor at run time of the command processor and interpreted by the command interpreter, wherein the configuration commands build graphical objects within the graphical user interface and assign functionality to the built graphical objects" and "one or more design tools corresponding to processes within an integrated circuit design process", "wherein the one or more design tools operate under control of the command processor and within the graphical user interface", as recited in claim 24. Instead, Nahaboo discloses that the interface designer uses an interface development tool to specify the interface, which tool is separate from the application with which the interface is operated. *See Nahaboo*, col. 1, lines 35-39, col. 2, lines 10-14 and 21-56.

Dangelo discloses that circuit specifications can be reused, but fails to overcome the deficiencies of Nahaboo. Instead, Dangelo discloses a "methodology for generating structural descriptions of complex digital devices from high-level descriptions and specifications," which "uses as systematic technique to map and enforce consistency of the semantics..." *See Dangelo*, Abstract. Further, Dangelo discloses "a Graphical User Interface (Graphical UI) 806 facilitates user interaction with the CDE by: abstracting out those steps of the [integrated circuit] design flow that do not require the designer's intervention, assisting and guiding the designer through the various stages of the design process as outlined by the synthesis framework, and assisting the designer in the composition of the constraints file for optimization." *See Dangelo*, col. 16, lines 35-42. In Dangelo, the building blocks of the circuit, such as wire traces, mega-cells, mega-function blocks, and the like are chosen from pre-designed building block libraries. *See, e.g., Dangelo*, col. 11, lines 1-10. Thus, Dangelo fails to disclose or suggest "the graphical user interface specified entirely by a user via a user input incuding one or more configuration commands...," as recited in claim 19.

Thus, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest "the graphical user interface specified entirely by a user via a user input including one or more configuration commands provided to the command processor at run time of the command processor and interpreted by the command interpreter, wherein the configuration commands build graphical objects within the graphical user interface and assign functionality to the built graphical objects" and "wherein the one or more design tools operate under control of the command processor and within the graphical user interface", as recited in claim 24, as recited in claim 19. Accordingly, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest at least one element of claim 24, and of claims 25-28, at least by virtue of their dependency from allowable claim 24.

C. CLAIMS 1 AND 5-7 ARE ALLOWABLE

Appellant respectfully traverses the rejection of claims 1 and 5-7 over the asserted combination of Nahaboo and Dangelo. In particular, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest all of the elements of the claims.

The asserted combination of Nahaboo and Dangelo fails to disclose or suggest "a command interpreter" that "modifies the graphical user interface at run time of the graphical user interface according to the interpreted configuration commands" including "building graphical objects within the graphical interface according to the interpreted configuration commands," "assigning functionality to the built graphical objects according to the interpreted configuration commands," and "displaying the graphical objects within the graphical interface according to the interpreted configuration commands," "wherein the graphical objects are selectable by a user to access the assigned functionality to produce an integrated circuit design", as recited in claim 1.

Nahaboo discloses a user interface description tool that uses an interpreted language where both data and the program have similar representations and that uses libraries of command objects and graphical objects. *See Nahaboo*, Abstract. Nahaboo notes that poor interface designs can result in mitigated commercial success. *See Nahaboo*, col. 1, lines 25-26.

Nahaboo discloses that "the purpose of this invention is to define an <u>extremely flexible</u>

<u>interface development tool</u> that can be used regardless of the application." *See Nahaboo*, col. 1, lines 29-34 (emphasis added). Nahaboo discloses that the interface development tool uses a list processing language (LISP) type of interpreted language that can be interpreted by an interpreter (WOOL) language program. *See Nahaboo*, col. 1, lines 35-39. Further, Nahaboo discloses that the interface development tool produces an interactive assembly geometry specification that is translated to a file that can be used by an application and stored on a disk, and that the file contains the user-interface format in the form of a WOOL language program. *See Nahaboo*, col. 2, lines 10-14.

Nahaboo states that the interactive interface tool allows the user to create and modify interfaces and windows that exhibit behaviors that can be easily duplicated. *See Nahaboo*, col. 4, lines 39-60. However, Nahaboo discloses an interface development tool to design and refine interfaces that can be used with applications, but does not suggest that the user of the applications allow for design and refinement of the interface "at run time of the graphical user interface," as recited in claim 1. Instead, Nahaboo discloses that the graphical user interface can be designed and edited and then stored for use with an application. At runtime of the application, the saved interface is loaded and executed using a WOOL program language interpreter that is embedded with the application. *See Nahaboo*, col. 1, lines 35-39. Thus, the WOOL interpreter program, **not the interface development tool,** is embedded with the application. While Nahaboo discloses that the interface description program can be expanded by adding other "widget" classes, Nahaboo discloses that "these classes can be added by describing the new attribute types of each class in the WOOL language so that the new "widgets" can be edited, for example, using the menu of the interface development tool. *See Nahaboo*, col. 2, lines 21-56. In Nahaboo, since the WOOL interpreter program loads and interprets the WOOL language program, such modifications or additions would take effect only upon restarting the application. Thus, Nahaboo discloses modification of the graphical user interface using an interface development tool, storage of the modified graphical user interface, and later execution using a WOOL interpreter program that is embedded with the application and that executes the stored graphical user interface file in conjunction with the application.

Applicant notes that Nahaboo discloses editing an interface using a separate application from that with which the interface is to be used. The edited interface is stored in a WOOL language program and the WOOL language interpreter is embedded with the application to interpret the interface file. However, Nahaboo does not disclose or suggest that the WOOL language interpreter "modifies the graphical user interface at run time of the graphical user interface", as recited in claim 1. Accordingly, Nahaboo does not disclose or suggest "a command interpreter" that "modifies the graphical user interface at run time of the graphical user interface", as recited in claim 1.

While Dangelo discloses that the schematic editors allow for selection of previously created and stored schematics, Dangelo does not disclose or suggest "a command interpreter" that "modifies the graphical user interface at run time of the graphical user interface", as recited in claim 1. In particular, Dangelo discloses a "methodology for generating structural descriptions of complex digital devices from high-level descriptions and specifications," which "uses as systematic technique to map and enforce consistency of the semantics..." *See Dangelo*, Abstract. Further, Dangelo discloses "a Graphical User Interface (Graphical UI) 806 facilitates user interaction with the CDE by: abstracting out those steps of the [integrated circuit] design flow that do not require the designer's intervention, assisting and guiding the designer through the various stages of the design process as outlined by the synthesis framework, and assisting the designer in the composition of the constraints file for optimization." *See Dangelo*, col. 16, lines 35-42. In Dangelo, the building blocks of the circuit, such as wire traces, mega-cells, mega-function blocks, and the like are chosen from pre-designed building block libraries. *See, e.g., Dangelo*, col. 11, lines 1-10.

Accordingly, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest all of the elements of claim 1, or of claims 5-7, at least by virtue of their dependency from allowable claim 1. Therefore, the rejection of claims 1 and 5-7 over the asserted combination of Nahaboo and Dangelo should be withdrawn.

D. CLAIMS 19-23 ARE ALLOWABLE

Claims 19-23 are allowable over the asserted combination of Nahaboo and Dangelo. In particular, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest "loading a configuration command of the one or more configuration commands into the command processor from at least one of a user specified command configuration script comprising the configuration command or from a command line in which the configuration command is entered by the user" and "assembling a graphical user interface having no hard coded objects and including at least one graphical user interface (GUI) component based on interpreted configuration commands from the user, the at least one graphical user interface (GUI) component selectable by a user to access an associated function to generate an integrated circuit design", as recited in claim 19. As discussed above, Nahaboo discloses that the interface development tool is separate from the application with which the interface is used, and that the developed interface is stored as a file with the application, which has an embedded WOOL processor to access and implement the interface file. *See Nahaboo*, col. 1, lines 35-39 and col. 2, lines 10-14 and 21-56.

However, Nahaboo does not disclose or suggest that that user of the application (as opposed to the interface design tool) in any way specifies the interface. Rather, Nahaboo discloses that the interface designer uses a development tool to specify the interface. *See Nahaboo*, col. 1, lines 35-39. Accordingly, Nahaboo does not disclose or suggest "assembling a graphical user interface having no hard coded objects and including at least one graphical user interface (GUI) component based on interpreted configuration commands from the user, the at least one graphical user interface (GUI) component selectable by a user to access an associated function to generate an integrated circuit design", as recited by claim 19.

Dangelo fails to overcome the deficiencies of Nahaboo. Instead, Dangelo discloses a "methodology for generating structural descriptions of complex digital devices from high-level descriptions and specifications," which "uses as systematic technique to map and enforce consistency of the semantics..." *See Dangelo*, Abstract. Further, Dangelo discloses "a Graphical User Interface (Graphical UI) 806 facilitates user interaction with the CDE by: abstracting out those steps of the [integrated circuit] design flow that do not require the designer's intervention, assisting and guiding the designer through the various stages of the design process as outlined by the synthesis framework, and assisting the designer in the composition of the constraints file for optimization." *See Dangelo*, col. 16, lines 35-42. In Dangelo, the building blocks of the circuit, such as wire traces, mega-cells, mega-function blocks, and the like are chosen from pre-designed building block libraries. *See, e.g., Dangelo*, col. 11, lines 1-10. While Dangelo discloses that circuit specifications can be reused, Dangelo does not disclose or suggest "assembling a graphical user interface having no hard coded objects and including at least one graphical user interface (GUI) component based on interpreted configuration commands from the user, the at least one graphical user interface (GUI) component selectable by a user to access an associated function to generate an integrated circuit design", as recited by claim 19.

Thus, the asserted combination of Nahaboo and Dangelo fails to disclose or suggest at least one element of claim 19, and of claims 20-23, at least by virtue of their dependency from allowable claim 19.

CONCLUSION

Applicants have specifically identified elements of the claims that are not disclosed or suggested by the cited references, alone or in combination, including Nahaboo and Dangelo. With this response, all of the pending claims 1, 5-16, and 19-28 are in condition for allowance. Reconsideration and notice to that effect is respectfully requested.

Respectfully submitted,

WESTMAN, CHAMPLIN & KELLY, P.A.

By: /R. Michael Reed/

R. Michael Reed, Reg. No. 59,066
900 Second Avenue South, Suite 1400
Minneapolis, Minnesota 55402-3319
Phone:(612) 334-3222 Fax:(612) 334-3312

RMR:amh

Appendix A

1. (Previously Presented) A command processor stored on a computer readable memory for use with a computer system comprising:

a graphical user interface program for providing a graphical interface to the computer system; and

a command interpreter, which loads one or more configuration commands into the command processor from at least one of a user specified command configuration script comprising the one or more configuration commands or from a command line interface in which the one or more configuration commands are entered by a user, interprets the configuration commands and modifies the graphical user interface at run time of the graphical user interface according to the interpreted configuration commands, including:

building graphical objects within the graphical interface according to the interpreted configuration commands;

assigning functionality to the built graphical objects according to the interpreted configuration commands; and

displaying the graphical objects within the graphical interface according to the interpreted configuration commands;

wherein the graphical objects are selectable by a user to access the assigned functionality to produce an integrated circuit design.

2-4. (Canceled)

5. (Original) The command processor of claim 1 and further comprising:

a suite of integrated circuit design tools, each design tool of the suite having a functionality corresponding to one or more steps in a design flow process of an integrated circuit.

6. (Previously Presented) The command processor of claim 5 wherein the command processor loads each design tool into the graphical user interface based on the user configuration commands.

7. (Previously Presented) The command processor of claim 1 and further comprising:
a graphics engine tool for drawing contents of a database into the graphical user interface based on the user configuration commands.

8. (Previously Presented) A method of providing a fully customizable graphical user interface comprising:
upon execution of a command processor by a user, loading a top level Tool Command Language (TCL) command into a namespace, the command processor including a graphical user interface (GUI) without graphical objects;
loading one or more TCL commands into the command processor from a command line in which the one or more TCL configuration commands are entered by the user;
building graphical objects according to the TCL configuration commands;
assigning functionality to the built graphical objects according to the TCL configuration commands; and
displaying the graphical objects within the GUI according to the TCL configuration commands, the graphical objects selectable by the user to produce an integrated circuit design.

9. (Original) The method of claim 8 and further comprising:
performing functions based on user interactions with the graphical objects according to their assigned functionality.

10. (Original) The method of claim 8 wherein the graphical objects are selected from a group consisting of windows, window panes, buttons, and menus.

11. (Previously Presented) The method of claim 8 wherein the step of assigning comprises:

creating the TCL command configuration script corresponding to a circuit design function; and

assigning the TCL command configuration script to one of the graphical objects.

12. (Original) The method of claim 11 wherein the one of the graphical objects is a button.

13. (Original) The method of claim 11 wherein the one of the graphical objects is an item within a pull-down menu.

14. (Original) The method of claim 8 and further comprising:

changing a look and feel of the graphical user interface during a circuit design process.

15. (Previously Presented) The method of claim 14 wherein the step of changing comprises:

creating new graphical objects, previously undefined by the command processor, using the TCL configuration commands; and

assigning functionality to the new graphical objects.

16. (Previously Presented) The method of claim 14 wherein the step of changing comprises:

loading a new top level TCL command into the namespace, which corresponds to one or more new TCL configuration commands;

building graphical objects according to the new TCL configuration commands;

assigning functionality to the built graphical objects according to the new TCL configuration commands; and

displaying the user-interactive window containing the graphical objects according to the new TCL configuration commands.

17. (Canceled)

18. (Canceled)

19. (Previously Presented) A method of providing a graphical user interface, the method comprising:

loading a top level Tool Command Language (TCL) command into a namespace upon execution of a command processor;

providing a command interpreter for interpreting one or more configuration commands from a user;

loading a configuration command of the one or more configuration commands into the command processor from at least one of a user specified command configuration script comprising the configuration command or from a command line in which the configuration command is entered by the user; and

assembling a graphical user interface having no hard coded objects and including at least one graphical user interface (GUI) component based on interpreted configuration commands from the user, the at least one graphical user interface (GUI) component selectable by a user to access an associated function to generate an integrated circuit design;

wherein all objects within the graphical user interface are user defined through the one or more configuration commands.

20. (Previously Presented) The method of claim 19 and further comprising:

changing the graphical user interface based on changing configuration commands from the user; and

displaying a changed graphical user interface during operation based on the changing configuration commands.

21. (Original) The method of claim 19 and further comprising:
interfacing with a suite of integrated circuit design tools for producing a integrated circuit layout and associated netlist.

22. (Original) The method of claim 21 wherein the step of interfacing comprises:
loading a design tool from the suite of design tools into the graphical user interface based on a user command.

23. (Previously Presented) The method of claim 22 wherein the one or more user configuration commands are assigned to one or more graphical objects.

24. (Previously Presented) An integrated circuit software design suite stored on a computer readable memory and comprising:
a command processor having a graphical user interface and a command interpreter for interpreting user commands, the graphical user interface specified entirely by a user via a user input including one or more configuration commands provided to the command processor at run time of the command processor and interpreted by the command interpreter, wherein the configuration commands build graphical objects within the graphical user interface and assign functionality to the built graphical objects;
one or more design tools corresponding to processes within an integrated circuit design process; and
wherein the one or more design tools operate under control of the command processor and within the graphical user interface.

25. (Previously Presented) The integrated circuit software design suite of claim 24, wherein the command processor interprets the user input to generate at least one graphical object within the graphical user interface associated with at least one design tool of the one or more design tools.

26. (Previously Presented) The integrated circuit software design suite of claim 25, wherein the graphical object is selectable by the user to load the at least one design tool into the graphical user interface, the graphical user interface accessible by the user to produce an integrated circuit design.

27. (Previously Presented) The command processor of claim 5, wherein at least one of the graphical objects is associated with at least one integrated design tool of the suite of integrated circuit design tools.

28. (Previously Presented) The command processor of claim 27, wherein the at least one integrated design tool is executable by the command interpreter to design and test an integrated circuit layout, and wherein the at least one of the graphical objects is selectable by the user to access the at least one integrated design tool.

EVIDENCE APPENDIX

None.

RELATED PROCEDINGS APPENDIX

There are no known related appeals or interferences that will directly affect or be directly affected by or have a bearing on the Board's decision in this Appeal.